MATLAB EXPO 2018

Automatización de Métodos y Procesos para Mejorar la Calidad del Diseño

Luis Lópe
Growing Complexity of Embedded Systems

McKendrick, J. “Cars become ‘datacenters on wheels’, carmakers become software companies,” ZDJNet, 2013
Why do 71% of Embedded Projects Fail?

Poor Requirements Management

Sources: Christopher Lindquist, Fixing the Requirements Mess, CIO Magazine, Nov 2005
Key Takeaways

- Author, manage requirements in Simulink
- Early verification to find defects sooner
- Automate manual verification tasks
- Workflow that conforms to safety standards

“Reduce costs and project risk through early verification, shorten time to market on a certified system, and deliver high-quality production code that was first-time right”  Michael Schwarz, ITK Engineering
Challenge with Traditional Development Process
Simulink Models for Specification
Complete Model Based Design

Simulink Models

Requirements → Executable Specification → Model used for production code generation → C/C++ → Generated code

Code Generation
Model Based Design Verification Workflow

Requirements → Executable Specification → Model used for production code generation → C/C++ → Generated code

- Component and system testing
- Review and static analysis
- Equivalence testing
- Equivalence checking

Simulink Models
Challenges with Requirements

Where are requirements implemented?

Is design and requirements consistent?

How are they tested?

Simulink Models

Requirements → Executable Specification → Model used for production code generation → C/C++ → Generated code
Gap Between Requirements and Design

- Requirements
- Executable Specification
- Model used for production code generation
- C/C++
- Generated code
Simulink Requirements

Author

If the Cancel switch is pressed, the value of `reqDrv` should be set to `reqMode.Cancel`.

Dashboard image:

Track

Enablements:

Manage

Issue: Destination Changed.

- Stored: Revision: 15
- Actual: Revision: 18

Clear Issue
To create a new requirement set to store requirements, click **New Requirement Set**. Save the requirement set to assign a name.

To add a requirement to a requirement set, select the requirement set and click **Add Requirement**. In the **Properties** pane, enter details for the requirement.

To add a child requirement, right-click a requirement and select **Add Child Requirement**.

To link a requirement to a block in your model, select the block, then right-click the requirement and select **Link from "object name" (object type)**. A link appears in the **Links** pane.

For information on linking using the Requirements Perspective, see **Getting Started** in the documentation.

To view a list of links, select **Links** from the **View** dropdown list in the toolstrip.

Change the source - destination relationship by selecting a link, and choosing a **Type** from the dropdown list in the **Properties** pane.
To create a new requirement set to store requirements, click **New Requirement Set**. Save the requirement set to assign a name.

To add a requirement to a requirement set, select the requirement set and click **Add Requirement**. In the **Properties** pane, enter details for the requirement.

To add a child requirement, right-click a requirement and select **Add Child Requirement**.

To link a requirement to a block in your model, select the block, then right-click the requirement and select **Link from "object name" (object type)**. A link appears in the **Links** pane.

For information on linking using the Requirements Perspective, see **Getting Started** in the documentation.

To view a list of links, select **Links** from the **View** dropdown list in the toolstrip.

Change the source - destination relationship by selecting a link, and choosing a **Type** from the dropdown list in the **Properties** pane.
Import Requirements from External Sources

Microsoft Word

Simulink Requirements Editor

**Import**

**3.1 Enabling cruise control**

Cruise control is enabled when the following conditions are met:

- Vehicle speed is within the target speed range (40km/h – 100km/h).
- Key position is ON.
- Gear position is Drive.
- Cruise button is pushed while the cruise control mode is disabled.

Dashboard image

**Keywords:**
- Revision information:
- Links
Requirements Perspective
Requirements Perspective
REQ.3.1 ENABLING CRUISE CONTROL
Cruise control is enabled when ....
REQ3.1 ENABLING CRUISE CONTROL
Cruise control is enabled when ….

ENABLE SWITCH DETECTION
If the Enable switch is pressed ….
Link Requirements, Designs and Tests

REQ.3.1 ENABLING CRUISE CONTROL
Cruise control is enabled when ..... 

ENABLE SWITCH DETECTION
If the Enable switch is pressed ..... 

Implemented By

reqMode.Cruise
REQ 3.1 Enabling Cruise Control
Cruise control is enabled when ..... 

Implemented By

Enable Switch Detection
If the Enable switch is pressed ..... 

Verified By

Test Case

Derives

reqMode.Cruise

MATLAB EXPO 2018
Track Implementation and Verification

![Image: Requirements - crs_controller]

**Implementation Status**
- Blue: Implemented
- Light Blue: Justified
- Missing

**Verification Status**
- Green: Passed
- Red: Failed
- Yellow: No Result
- Missing

**Verification Status**
- Green: Passed
- Red: Failed
- Yellow: No Result
- Missing
Original Requirement
If the switch is pressed and the counter reaches \( 50 \) then it shall be recognized as a long press of the switch.

Updated Requirement
If the switch is pressed and the counter reaches \( 75 \) then it shall be recognized as a long press of the switch.
Verify Design to Guidelines and Standards

Is the design built right?

Is it too complex?

Is it ready for code generation?

Review and static analysis

Simulink Models

Requirements → Executable Specification → Model used for production code generation → C/C++ → Generated code

MATLAB EXPO 2018
Automate verification with static analysis

Requirements
Executable Specification
Model used for production code generation
C/C++
Generated code

Model Advisor Analysis

Check for:
- Readability and Semantics
- Performance and Efficiency
- Clones
- And more……
Generate reports for reviews and documentation

Model Advisor Analysis

Model Advisor Reports

Simulink Models

Requirements

Executable Specification

Model used for production code generation

C/C++

Generated code
Navigate to Problematic Blocks

<table>
<thead>
<tr>
<th>Block</th>
<th>Block Type</th>
<th>Code generation support</th>
<th>Recommendation for C/C++ production code deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td>../../Intake Manifold/p0 = 0.589 bar</td>
<td>Integrator</td>
<td>Yes(^2)</td>
<td>No</td>
</tr>
<tr>
<td>sldemo_fuelsys/Throttle Command</td>
<td>Repeating table</td>
<td>Yes(^3)</td>
<td>No</td>
</tr>
</tbody>
</table>

Simulink Models

- Requirements
- Executable Specification
- Model used for production code generation
- C/C++
  - Generated code

RT/VM

\(0.41328\)

\(1\) s

\(p_0 = 0.569\) bar

\(2\) rad/s

N (rad/sec)
Recommended Action
Although Embedded Coder supports these blocks, they are not recommended for C/C++ production code deployment. Review the support notes for these blocks and follow the given advice.
Built in checks for industry standards and guidelines

- DO-178/DO-331
- ISO 26262
- IEC 61508
- IEC 62304
- EN 50128
- MISRA C:2012
- CERT C, CWE, ISO/IEC TS 17961
- MAAB (MathWorks Automotive Advisory Board)
- JMAAB (Japan MATLAB Automotive Advisory Board)
Configure and customize analysis

Simulink Models

Requirements → Executable Specification → Model used for production code generation → C/C++ → Generated code
Detect Design Errors with Formal Methods

- Find run-time design errors:
  - Integer overflow
  - Dead Logic
  - Division by zero
  - Array out-of-bounds
  - Range violations

- Generate counter example to reproduce error

Executable Specification

Model used for production code generation

Generated code
Prove That Design Meets Requirements

- Prove design properties using formal requirement models
- Model functional and safety requirements
- Generates counter example for analysis and debugging
Checks for standards and guidelines are often performed late.
Shift Verification Earlier With Edit-Time Checking

- Highlight violations as you edit
- Fix issues earlier
- Avoid rework
Find Compliance Issues as you Edit with Edit-Time Checking
Assess Quality with Metrics Dashboard

- Consolidated view of metrics
  - Size
  - Compliance
  - Complexity

- Identify where problem areas may be
Grid Visualization for Metrics

- Visualize Standards
- Check Compliance
  - Find Issues
  - Identify patterns
  - See hot spots

Legend:
- Red: Fail
- Orange: Warning
- Green: Pass
- Gray: Not run
Functional Testing

- Does the design meet requirements?
- Is it functioning correctly?
- Is it completely tested?

Simulink Models

Requirements → Executable Specification → Model used for production code generation → C/C++ → Generated code
Systematic Functional Testing

Test Case

Inputs
- MAT file (input)
  - Group 1
    - Signal 1
  - Signal Builder
  - Test Sequence
  - and more!

Test Harness

Main Model

Assessments
- MAT file (baseline)
- MATLAB Unit Test
  - Test Assessment
  - and more!

Excel file (input)

Excel file (baseline)

MATLAB EXPO 2018
Manage Testing and Test Results
Coverage Analysis to Measure Testing

- Identify testing gaps
- Missing requirements
- Unintended Functionality
- Design Errors
Test Case Generation for Functional Testing

- Specify functional test objectives
  - Define custom objectives that signals must satisfy in test cases

- Specify functional test conditions
  - Define constraints on signal values to constrain test generator
Static Code Analysis

Is the code compliant to MISRA?

Is integrated code free of run-time errors?

Is interface between generated and other code fully tested?

The Generated Code is integrated with Other Code (Handwritten)
Static Code Analysis with Polyspace

- **Code metrics and standards**
  - Comment density, cyclomatic complexity,…
  - MISRA and Cybersecurity standards
  - Support for DO-178, ISO 26262, ….

- **Bug finding and code proving**
  - Check data and control flow of software
  - Detect bugs and security vulnerabilities
  - Prove absence of runtime errors

Results from Polyspace Code Prover
Equivalence Testing

Is the code functionally equivalent to model?

Is all the code tested?

Simulink Models

Requirements → Executable Specification → Model used for production code generation → C/C++ → Generated code
Equivalence Testing

- **Software in the Loop (SIL)**
  - Show functional equivalence, model to code
  - Execute on desktop / laptop computer

- **Processor in the Loop (PIL)**
  - Numerical equivalence, model to target code
  - Execute on target board

- **Re-use tests developed for model to test code**
- **Collect code coverage**
Qualify tools with IEC Certification Kit and DO Qualification Kit

- Qualify code generation and verification products
- Includes documentation, test cases and procedures

KOSTAL Asia R&D Center Receives ISO 26262 ASIL D Certification for Automotive Software Developed with Model-Based Design

BAE Systems Delivers DO-178B Level A Flight Software on Schedule with Model-Based Design

Kostal’s electronic steering column lock module.

Primary flight control computers from BAE Systems.
Lear Delivers Quality Body Control Electronics Faster Using Model-Based Design

Challenge
Design, verify, and implement high-quality automotive body control electronics

Solution
Use Model-Based Design to enable early and continuous verification via simulation, SIL, and HIL testing

Results
- Requirements validated early. Over 95% of issues fixed before implementation, versus 30% previously
- Development time cut by 40%. 700,000 lines of code generated and test cases reused throughout the development cycle
- Zero warranty issues reported

“We adopted Model-Based Design not only to deliver better-quality systems faster, but because we believe it is a smart choice. Recently we won a project that several of our competitors declined to bid on because of its tight time constraints. Using Model-Based Design, we met the original delivery date with no problem.”
- Jason Bauman, Lear Corporation

MATLAB Expo 2018

Link to user story
Customer References and Applications

Airbus Helicopters Accelerates Development of DO-178B Certified Software with Model-Based Design
Software testing time cut by two-thirds

LS Automotive Reduces Development Time for Automotive Component Software with Model-Based Design
Specification errors detected early

Continental Develops Electronically Controlled Air Suspension for Heavy-Duty Trucks
Verification time cut by up to 50 percent

More User Stories: www.mathworks.com/company/user_stories.html
Summary

1. Author and manage requirements within Simulink
2. Find defects earlier
3. Automate manual verification tasks
4. Reference workflow that conforms to safety standards
Learn More

Visit MathWorks Verification, Validation and Test Solution Page: mathworks.com/solutions/verification-validation.html
Thank You!